CLAIMS

We claim:

1. A method for detecting software development best practice violations, comprising:

receiving sets of source code from a plurality of sources;

extracting at least one code pattern from the sets of source code;

defining meta data for each of the at least one code pattern that indicates a quality of the at least one code pattern; and

assigning a rank to each of the at least one code pattern based on the corresponding meta data.

2. The method of claim 1, wherein the rank is further based on a skill level and an experience level of a developer of the at least one code pattern.

3. The method of claim 1, further comprising determining a programming language of the sets of source code.

4. The method of claim 3, wherein the meta data for each of the at least one code pattern identifies the programming language, a list of most used classes, a list of dependencies, a number and a type of objects created and used at run time, and memory usage of the at least one code pattern.

5. The method of claim 1, wherein the plurality of sources comprises a plurality of nodes interconnected in a peer-to-peer network environment, and wherein each of the plurality of nodes is operated by a developer.

6. The method of claim 5, further comprising registering the developers, prior to the receiving step.

7. The method of claim 6, wherein the registering step comprises collecting contact information, a skill level and an experience level corresponding to the developers.

8. The method of claim 7, wherein the registering step further comprises collecting feedback information about each developer from the other developers.

9. The method of claim 1, further comprising:

receiving a subsequent set of source code;

extracting and classifying a code pattern to be tested from the subsequent set of source code;

comparing the code pattern to be tested to the at least one code pattern to determine a closest match to the code pattern to be tested;

assigning a rank of the closest match to the code pattern to be tested; and

detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

10. A method for building a dynamic best practice violation (BPV) engine resource for detecting software development best practice violations, comprising:

receiving sets of source code from a plurality of sources;

detecting a programming language of each of the sets of source code;

extracting a plurality of code patterns from the sets of source code;

defining meta data for each of the plurality of code patterns that indicates a quality of the plurality of code patterns; and

classifying and assigning a rank to each of the plurality of code patterns based on the corresponding meta data.

11. The method of claim 10, wherein the plurality of sources comprises a plurality of nodes interconnected in a peer-to-peer network environment, and wherein each of the plurality of nodes is operated by a developer.

12. The method of claim 11, further comprising registering the developers, prior to the receiving step.

13. The method of claim 12, wherein the registering step comprises collecting contact information, a skill level and an experience level corresponding to the developers.

14. The method of claim 13, wherein the registering step further comprises collecting feedback information about each developer from the other developers.

15. The method of claim 10, wherein the rank is further assigned to each of the plurality of code patterns based a skill level and a experience level of a developer of each of the sets of source code.

16. The method of claim 10, wherein the meta data for each of the plurality of code patterns identifies the programming language, a list of most used classes, a list of dependencies, a number and a type of objects created and used at run time, and memory usage of each of the plurality of code patterns.

17. The method of claim 10, further comprising:

receiving a subsequent set of source code;

extracting and classifying a code pattern to be tested from the subsequent set of source code;

comparing the code pattern to be tested to the plurality of code patterns to determine a closest match to the code pattern to be tested;

assigning a rank of the closest match to the code pattern to be tested; and

detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

18. A method for detecting software development best practice violations, comprising:

receiving a first set of source code in a best practice violation (BPV) engine;

extracting and classifying a code pattern to be tested from the first set of source code;

comparing the code pattern to be tested to a plurality of code patterns extracted from other sets of source code previously received and analyzed by the BPV engine to determine a closest match to the code pattern to be tested;

assigning a rank previously assigned to the closest match to the code pattern to be tested; and

detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

19. The method of claim 18, further comprising building a BPV engine resource prior to the receiving step, wherein the building step comprises:

receiving the other sets of source code from a plurality of sources;

detecting a programming language of each of the other sets of source code;

extracting the plurality of code patterns from the sets of source code;

defining meta data for each of the plurality of code patterns that indicates a quality of the plurality of code patterns; and

assigning a rank to each of the plurality of code patterns based on the corresponding meta data.

20. The method of claim 19, wherein the rank is further assigned to each of the plurality of code patterns based a skill level and a experience level of a developer of each of the other sets of source code.

21. The method of claim 19, wherein the plurality of sources comprises a plurality of nodes interconnected in a peer-to-peer network environment, and wherein each of the plurality of nodes is operated by a developer.

22. The method of claim 21, further comprising registering the plurality of nodes, prior to the receiving step.

23. The method of claim 22, wherein the registering step comprises collecting contact information, a skill level and an experience level corresponding to the developers.

24. The method of claim 23, wherein the registering step further comprises collecting feedback information about each developer from the other developers.

25. A system for building a dynamic best practice violation (BPV) engine resource for detecting software development best practice violations, comprising:

a code reception system for receiving sets of source code from a plurality of sources;

a language detection system for detecting a programming language of each of the sets of source code;

a pattern extraction system for extracting a plurality of code patterns from the sets of source code;

a code pattern analysis system for defining meta data for each of the plurality of code patterns that indicates a quality of the plurality of code patterns; and

a classification and ranking system for classifying and assigning a rank to each of the plurality of code patterns based on the corresponding meta data.

26. The system of claim 25, wherein the plurality of sources comprises a plurality of nodes interconnected in a peer-to-peer network environment, and wherein each of the plurality of nodes is operated by a developer.

27. The system of claim 26, further comprising a registration system for registering the developers, prior to the receiving step.

28. The system of claim 27, wherein the registration system collects contact information, a skill level and an experience level corresponding to the developers.

29. The system of claim 28, wherein the registration system collects feedback information about each developer from the other developers.

30. The system of claim 25, wherein the rank is further assigned to each of the plurality of code patterns based a skill level and a experience level of a developer of each of the sets of source code.

31. The system of claim 25, wherein the meta data for each of the plurality of code patterns identifies the programming language, a list of most used classes, a list of dependencies, a number and a type of objects created and used at run time, and memory usage of each of the plurality of code patterns.

32. The system of claim 25, further comprising a BPV engine comprising:

a test reception system for receiving a subsequent set of source code;

a test extraction system for a code pattern to be tested from the subsequent set of source code;

a test classification system for classifying the code pattern to be tested

a matching system for comparing the code pattern to be tested to the plurality of code patterns to determine a closest match to the code pattern to be tested, and for assigning a rank of the closest match to the code pattern to be tested; and

a deviation detection system for detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

33. The system of claim 32, wherein the BPV engine further comprises a recommendation system for recommending at least one alternative to the code pattern to be tested if the deviation is detected.

34. A best practice violation (BPV) engine for detecting software development best practice violations, comprising:

a test reception system for receiving a first set of source code;

a test extraction system for extracting a code pattern to be tested from the first set of source code;

a test classification system for classifying the code pattern to be tested;

a matching system for comparing the code pattern to be tested to a plurality of code patterns extracted from other sets of source code previously received and analyzed by the BPV engine to determine a closest match to the code pattern to be tested, and for assigning a rank previously assigned to the closest match to the code pattern to be tested; and

a deviation detection system for detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

35. The BPV engine of claim 34, further comprising a recommendation system for recommending at least one alternative to the code pattern to be tested if the deviation is detected.

36. The BPV engine of claim 34, wherein the rank of the closest match is determined based on a quality of the closest match, and a skill level and an experience level of a developer of the closest match.

37. A program product stored on a recordable medium for building a dynamic best practice violation (BPV) engine resource for detecting software development best practice violations, which when executed, comprises:

program code for receiving sets of source code from a plurality of sources;

program code for detecting a programming language of each of the sets of source code;

program code for extracting a plurality of code patterns from the sets of source code;

program code for defining meta data for each of the plurality of code patterns that indicates a quality of the plurality of code patterns; and

program code for classifying and assigning a rank to each of the plurality of code patterns based on the corresponding meta data.

38. The program product of claim 37, wherein the plurality of sources comprises a plurality of nodes interconnected in a peer-to-peer network environment, and wherein each of the plurality of nodes is operated by a developer.

39. The program product of claim 38, further comprising program code for registering the developers, prior to the receiving step.

40. The program product of claim 39, wherein the program code for registering collects contact information, a skill level and an experience level corresponding to the developers.

41. The program product of claim 40, wherein the program code for registering collects feedback information about each developer from the other developers.

42. The program product of claim 37, wherein the rank is further assigned to each of the plurality of code patterns based a skill level and a experience level of a developer of each of the sets of source code.

43. The program product of claim 37, wherein the meta data for each of the plurality of code patterns identifies the programming language, a list of most used classes, a list of dependencies, a number and a type of objects created and used at run time, and memory usage of each of the plurality of code patterns.

44. The program product of claim 37, further comprising a BPV engine comprising:

program code for receiving a subsequent set of source code;

program code for a code pattern to be tested from the subsequent set of source code;

program code for classifying the code pattern to be tested

program code for comparing the code pattern to be tested to the plurality of code patterns to determine a closest match to the code pattern to be tested, and for assigning a rank of the closest match to the code pattern to be tested; and

program code for detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

45. The program product of claim 44, wherein the BPV engine further comprises program code for recommending at least one alternative to the code pattern to be tested if the deviation is detected.

46. A best practice violation (BPV) engine stored on a recordable medium for detecting software development best practice violations, which when executed, comprises:

program code for receiving a first set of source code;

program code for extracting a code pattern to be tested from the first set of source code;

program code for classifying the code pattern to be tested;

program code for comparing the code pattern to be tested to a plurality of code patterns extracted from other sets of source code previously received and analyzed by the BPV engine to determine a closest match to the code pattern to be tested, and for assigning a rank previously assigned to the closest match to the code pattern to be tested; and

program code for detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

47. The BPV engine of claim 46, further comprising program code for recommending at least one alternative to the code pattern to be tested if the deviation is detected.

48. The BPV engine of claim 46, wherein the rank of the closest match is determined based on a quality of the closest match, and a skill level and an experience level of a developer of the closest match.

49. A computer-implemented business method for detecting software development best practice violations, comprising:

receiving a first set of source code in a best practice violation (BPV) engine from a subscriber;

extracting and classifying a code pattern to be tested from the first set of source code;

comparing the code pattern to be tested to a plurality of code patterns extracted from other sets of source code previously received and analyzed by the BPV engine to determine a closest match to the code pattern to be tested;

assigning a rank previously assigned to the closest match to the code pattern to be tested; and

detecting a software development best practice violation if the rank assigned to the code pattern to be tested fails to comply with a predetermined threshold.

50. The method of claim 49, further comprising recommending at least one alternative for the code pattern to be tested to the subscriber if the code pattern to be tested fails to comply with the predetermined threshold.

51. The method of claim 49, wherein the rank previously assigned to the closest match is assigned is based on a quality of the closest match. and a skill level and an experience level of a developer of the closest match.

52. The method of claim 49, wherein the method is offered as a fee-based service to the subscriber.